# 5 TEST MANAGEMENT

Geoff Thompson

## INTRODUCTION

This chapter provides a generic overview of how testing is organised and how testing is managed within organisations. A generic view of testing will, inevitably, not match the way testing is organised in specific organisations. The issues addressed are nevertheless important for any organisation and need to be considered by all.

We will start by looking at how testing and risk fit together, as well as providing detailed coverage of test planning and the control of testing, and we will identify how independence assists the test process. One very important area in managing the test process is the understanding of the different roles and tasks associated with the testing role such as the test leader and the tester.

We cannot, in one chapter, provide all the knowledge required to enable the reader to become a practising test leader or test manager, but we do aim to provide the background information necessary for a reader to understand the various facets of the test management role.

### Learning objectives

The learning objectives for this chapter are listed below. You can confirm that you have achieved these by using the self-assessment questions at the start of the chapter, the 'Check of understanding' boxes distributed throughout the text, and the example examination questions provided at the end of the chapter. The chapter summary will remind you of the key ideas.

The sections are allocated a K number to represent the level of understanding required for that section; where an individual element has a lower K number than the section as a whole this is indicated for that topic; for an explanation of the K numbers see the Introduction.

#### Test organisation (K2)

- Recognise the importance of independent testing. (K1)
- Explain the benefits and drawbacks of independent testing within an organisation.

- Recognise the different team members to be considered for the creation of a test team. (K1)
- Recall the tasks of typical test leader and tester. (K1)

### Test planning and estimation (K3)

- Recognise the different levels and objectives of test planning. (K1)
- Summarise the purpose and content of the test plan, test design specification and test procedure documents according to the *Standard for Software Test Documentation* (IEEE 829-1998). (K2)
- Differentiate between conceptually different test approaches, such as analytical model based methodical, process/standard compliant, dynamic/heuristic, consultative and regression averse. (K2)
- Differentiate between the subject of test planning for a system and for scheduling test execution. (K2)
- Write a test execution schedule for a given set of test cases, considering prioritisation, and technical and logical dependencies.
- List test preparation and execution activities that should be considered during test planning. (K1)
- Recall typical factors that influence the effort related to testing. (K1)
- Differentiate between two conceptually different estimation approaches: the metrics-based approach and the expert-based approach. (K2)
- Recognise/justify adequate exit criteria for specific test levels and groups of test cases (e.g. for integration testing, acceptance testing or test cases for usability testing). (K2)

### Test progress monitoring and control (K2)

- Recall common metrics used for monitoring test preparation and execution. (K1)
- Explain and compare test metrics for test reporting and test control (e.g. defects found and fixed, and tests passed and failed) related to purpose and use.
- Summarise the purpose and content of the test summary report document according to the *Standard for Software Test Documentation* (IEEE 829-1998).

### Configuration management (K2)

- Summarise how configuration management supports testing.

### Risk and testing (K2)

- Describe a risk as a possible problem that would threaten the achievement of one or more stakeholders' project objectives.

- Remember that risks are determined by likelihood (of happening) and impact (harm resulting if it does happen). (K1)
- Distinguish between the project and product risks.
- Recognise typical product and project risks. (K1)
- Describe, using examples, how risk analysis and risk management may be used for test planning.

### Incident management (K3)

- Recognise the content of an incident report according to the *Standard for Software Test Documentation* (IEEE 829-1998). (K1)
- Write an incident report covering the observation of a failure during testing.

## Self-assessment questions

The following questions have been designed to enable you to check your current level of understanding for the topics in this chapter. The answers are given at the end of the chapter.

### Question SA1 (K1)

What is the purpose of exit criteria?

a. To identify how many tests to design.
b. To identify when to start testing.
c. To identify when to stop testing.
d. To identify who will carry out the test execution.

### Question SA2 (K2)

Which of the following are **most** likely to be used when developing a test strategy or test approach?

(i) Failure-based approach
(ii) Test specification approach
(iii) Model-based approach
(iv) Finance-based approach

a. (iii) and (ii)
b. (i) and (iv)
c. (ii) and (i)
d. (i) and (iii)

### Question SA3 (K1)

What can a risk-based approach to testing provide?

a. The types of test techniques to be employed.
b. The total tests needed to provide 100 per cent coverage.
c. An estimation of the total cost of testing.
d. Only that test execution is effective at reducing risk.

## RISK AND TESTING

It is not possible to talk about test management without first looking at risk and how it affects the fundamental test process defined in Chapter 2. If there were no risk of adverse future events in software or hardware development then there would be no need for testing. In other words, if defects did not exist then neither would testing.

Risk can be defined as the chance of an event, hazard, threat or situation occurring and its undesirable consequences:

> **Risk – a factor that could result in future negative consequences, usually expressed as impact and likelihood.**

In a project a test leader will use risk in two different ways: project risks and product risks. In both instances the calculation of the risk will be:

> **Level of risk = probability of the risk occurring × impact if it did happen**

### Project risks

Whilst managing the testing project a test leader will use project risks to manage the capability to deliver.

Project risks include:

- Supplier issues:
  - Failure of a third party to deliver on time or at all.
  - Contractual issues, such as meeting acceptance criteria.
- Organisational factors:
  - Skills, training and staff shortages.
  - Personal issues.
  - Political issues, such as problems that stop testers communicating their needs and test results.
  - Failure by the team to follow up on information found in testing and reviews (e.g. not improving development and testing practices).
  - Improper attitude toward or expectations of testing (e.g. not appreciating the value of finding defects during testing).

- Technical issues:
  - Problems in defining the right requirements.
  - The extent that requirements can be met given existing project constraints.
  - Test environment not ready on time.
  - Late data conversion, migration planning and development and testing data conversion/migration tools.
  - Low quality of the design, code, configuration data, test data and tests.

For each risk identified a probability (chance of the risk being realised) and impact (what will happen if the risk is realised) should be identified as well as the identification and management of any mitigating actions (actions aimed at reducing the probability of a risk occurring, or reducing the impact of the risk if it did occur).

So, for example if there was a risk identified that the third-party supplier may be made bankrupt during the development, the test manager would review the supplier's accounts and might decide that the probability of this is medium (3 on a scale of 1 to 5, 1 being a high risk and 5 a low one). The impact on the project if this did happen would be very high (1 using the same scale). The level of risk is therefore 3 × 1 = 3. Thus, the lower the number, the more the risk.With 3 being in the medium risk area the test leader would now have to consider what mitigating actions to take to try to stop the risk becoming a reality. This might include not using the third party, or ensuring that payment for third-party deliverables is made efficiently.

When analysing, managing and mitigating these risks the test manager is following well-established project management principles provided within project management methods and approaches. The project risks recognised during test planning should be documented in the IEEE 829 test plan (see later in this chapter for details of the test plan contents); for the ongoing management and control of existing and new project risks a risk register should be maintained by the test leader.

**Product risks**

When planning and defining tests a test leader or tester using a risk-based testing approach will be managing product risks.

Potential failure areas (adverse future events or hazards) in software are known as product risks, as they are a risk to the quality of the product. In other words, the potential of a defect occurring in the live environment is a product risk. Examples of product risks are:

- Failure-prone software delivered.
- The potential that a defect in the software/hardware could cause harm to an individual or company.

- Poor software characteristics (e.g. functionality, security, reliability, usability, performance).
- Poor data integrity and quality (e.g. data migration issues, data conversion problems, data transport problems, violation of data standards).
- Software that does not perform its intended functions.

Risks are used to decide where to start testing in the software development life cycle, e.g. the risk of poor requirements could be mitigated by the use of formal reviews as soon as the requirements have been documented at the start of a project. Product risks also provide information enabling decisions regarding how much testing should be carried out on specific components or systems, e.g. the more risk there is, the more detailed and comprehensive the testing may be. In these ways testing is used to reduce the risk of an adverse effect (defect) occurring or being missed.

Mitigating product risks may also involve non-test activities. For example, in the poor requirements situation, a better and more efficient solution may be simply to replace the analyst who is writing the poor requirements in the first place.

As already stated, a risk-based approach to testing provides proactive opportunities to reduce the levels of product risk starting in the initial stages of a project. It involves the identification of product risks and how they are used to guide the test planning, specification and execution. In a risk-based approach the risks identified:

- will determine the test techniques to be employed, and/or the extent of testing to be carried out, e.g. the Motor Industry Software Reliability Association (MISRA) defines which test techniques should be used for each level of risk: the higher the risk, the higher the coverage required from test techniques;
- prioritise testing in an attempt to find the critical defects as early as possible, e.g. by identifying the areas most likely to have defects (the most complex) the testing can be focused on these areas;
- will determine any non-test activities that could be employed to reduce risk, e.g. to provide training to inexperienced designers.

Risk-based testing draws on the collective knowledge and insights of the project stakeholders, testers, designers, technical architects, business reps and anyone with knowledge of the solution to determine the risks and the levels of testing to address those risks.

To ensure that the chance of a product failure is minimised, risk management activities provide a disciplined approach:

- To assess continuously what can go wrong (risks). Regular reviews of the existing and looking for any new product risks should occur periodically throughout the life cycle.

- To determine what risks are important to deal with (probability × impact). As the project progresses, owing to the mitigation activities risks may reduce in importance, or disappear altogether.
- To implement actions to deal with those risks (mitigating actions).

Testing supports the identification of new risks by continually reviewing risks of the project deliverables throughout the life cycle; it may also help to determine what risks are important to reduce by setting priorities; it may lower uncertainty about risks by, for example, testing a component and verifying that it does not contain any defects; and lastly by running specific tests it may verify other strategies that deal with risks, such as contingency plans.

Testing is a risk control activity that provides feedback about the residual risk in the product by measuring the effectiveness of critical defect removal (see below) and by reviewing the effectiveness of contingency plans.

**CHECK OF UNDERSTANDING**

(1) What are the two types of risks that have to be considered in testing?

(2) Compare and contrast these two risk types.

(3) How early in the life cycle can risk impact the testing approach?

(4) What does MISRA determine when the level of risk is understood?

## TEST ORGANISATION

### Test organisation and independence

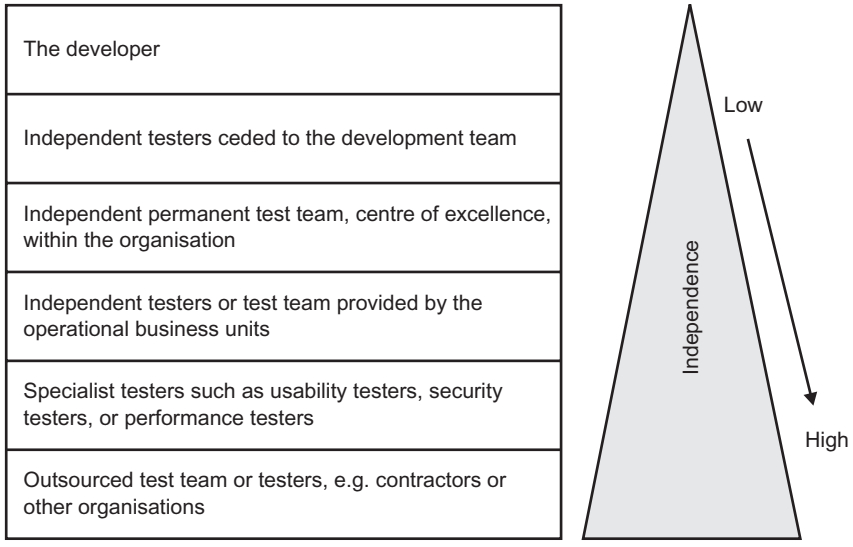
Independent testing is testing carried out by someone other than the creator (developer) of the code being tested. By remaining independent it is possible to improve the effectiveness of testing if implemented correctly.

As humans we are all capable of making mistakes, from the simplest misspelling or wrong use of syntax to fundamental errors at the core of any documents we write. The problem is that as authors we are less able to see our own errors than someone else, who is less directly associated with the document, would be. This is a problem that is made worse, in the world of software development, by the differing 'world view' of testers and developers. A developer, as the creator and owner of documents and code related to development, perceives these deliverables as being correct when they are delivered. The general awareness that we all make mistakes is, at this stage, overridden by the belief that what has been produced is what is required. A tester, by contrast, will take the view that anything delivered for testing is likely to contain errors and will search diligently to identify and locate those errors.

This is where independent testing is important, as it is genuinely hard for authors to identify their own errors, but it is easier for others to see them. There are many options for many levels of independence. In general, the more remote a tester is from the production of the document, the greater is the level

of independence. Figure 5.1 indicates the most common roles and the levels of independence they bring.

**Figure 5.1** Levels of independent testing




Of course independence comes at a price. The greater the level of independence, the greater the likelihood of errors in testing arising from unfamiliarity. Levels of independence will also depend on the size of the organisation. In smaller organisations where everybody contributes to every activity it is harder to differentiate the role of the tester from any other role, and therefore testers may not be very independent at all. The key in these circumstances is for the testers to have independence of mind, not necessarily to be in an independent (separate) team. In organisations where there are clearly defined roles it is a lot easier for a tester to remain independent.

It is also possible to mix and match the levels of independence, e.g. a test team made up of permanent resources, business unit resources and contractors. For large, complex or safety-critical projects, it is usually best to have multiple levels of testing, with some or all of the levels done by independent testers.

The 'agile' approach to development challenges the traditional approach to independence. In this approach everybody takes on multiple roles and so maintaining total independence is not always possible. A tester in this situation has to be able to switch to an independent view, at the relevant points in the project. Testers achieve this independence of view by not assuming anything and by not starting to own the software like a developer would, e.g. the view that was how it was developed to work.

Independence in the implementation of testing has some key benefits and drawbacks, as in Table 5.1.

**Table 5.1** Features of independent testing

| Benefits | Drawbacks |
|---|---|
| The tester sees other and different defects to the author | Isolation from the development team (if treated as totally independent), which will mean the tester is totally dependent on the test basis to understand what it is the tester is testing (documentation that is rarely up to date) |
| The tester is unbiased | The tester may be seen as the bottleneck, as independent test execution is normally the last stage and affected by any delays earlier in the process |
| The tester can see what has been built rather than what the developer thought had been built | Developers lose a sense of responsibility for quality as it may be assumed that they need not worry about errors because the independent test team will find them |
| The tester makes no assumptions regarding quality | The fully independent view sets developers and testers on either side of an invisible fence. This can be a hindrance to communication that would in normal circumstances ensure common understanding and effective working. It can also mean that developers are seen to 'throw' the software over the fence |

**CHECK OF UNDERSTANDING**

(1) Why is independent testing more effective at finding errors than simply allowing the developer and author to test their own product?

(2) Name three benefits of independence.

(3) Which organisation provides the lowest level of independence and which provides the highest?

### Tasks of a test leader and tester

Test tasks are traditionally carried out by people who make testing a career; however, test tasks may also be carried out by non-testers such as a project manager, quality manager, developer, business and domain expert, infrastructure or IT operations. The availability of resources usually determines the resource types that are deployed on each project, e.g. if there are no career testers available an organisation may identify non-testing IT or business resources to carry out the role of tester for a specific project or time period.

The syllabus defines two testing roles, the test leader (or test manager/test coordinator) and the tester. Other roles may exist in your organisation, but they are not covered here.

The testing roles can be undertaken by anyone with the required skills or who is given the right training. For example, the role of a test leader could be undertaken by a project manager. The decision as to who does what will depend on how a project or organisation is structured, as well as the size and number of resources working on a given project.

It is important to understand here the difference between a testing role and a testing job. A role is an activity, or a series of activities given to a person to fulfil, e.g. the role of test leader. A person may therefore have more than one role at any moment depending on their experience and the level of workload on a project. A job is effectively what an individual is employed to do, so one or many roles could make up a job. For example, a test leader could also be a tester.

The tasks undertaken by a test leader align very closely with those undertaken by a project manager and align closely with standard approaches to project management. In this context a test leader is anyone who leads a team of testers (be that one or many testers). They are also known as test programme managers, test managers, test team leaders and test coordinators.

Typical test leader tasks may include:

- Coordinating the development of the test strategy and plan with project managers and others.
- Writing or reviewing test strategies produced for the project, and test policies produced for the organisation.
- Contributing the testing perspective to other project activities, such as development delivery schedules.
- Planning the development of the required tests – which will include ensuring that the development uses the correct understanding of the risks, selecting the required test approaches (test levels, cycles, approach, objectives and incident management planning), estimating the time and effort and converting to the cost of testing and acquiring the right resources.

- Managing the specification, preparation, implementation and execution of tests, including the monitoring and control of all the specification and execution.
- Taking the required action, including adapting the planning, based on test results and progress (sometimes documented in status reports), and any action necessary to compensate for problems or delays.
- Ensuring that adequate configuration management of testware is in place and that the testware is fully traceable, e.g. there is a hierarchical relationship established between the requirements and the detailed specification documents.
- Putting in place suitable metrics for measuring test progress and evaluating the quality of the testing delivered and the product.
- Agreeing what should be automated, to what degree, and how, ensuring it is implemented as planned.
- Where required, selecting tools to support testing and ensuring any tool training requirements are met.
- Agreeing the structure and implementation of the test environment.
- Scheduling all testing activity.
- At the end of the project, writing a test summary report based on the information gathered during testing.

These tasks are not, however, all of the tasks that could be carried out by test leaders, just the most common ones. In fact other resources could take on one or more of these tasks as required, or they may be delegated to other resources by the test leader. The key is to ensure that everyone is aware of who is doing what tasks, that they are completed on time and within budget, and that they are tracked through to completion.

The other role covered by the syllabus is that of the tester, also known as test analyst or test executor.

The tasks typically undertaken by a tester may include:

- Reviewing and contributing to the development of test plans.
- Analysing, reviewing and assessing user requirements, specifications and models for testability.
- Creating test specifications from the test basis.
- Setting up the test environment (often coordinating with system administration and network management). In some organisations the setting up and management of the test environment could be centrally controlled; in this situation a tester would directly liaise with the environment management to ensure the test environment is delivered on time and to specification.

- Preparing and acquiring/copying/creating test data.
- Implementing tests on all test levels, executing and logging the tests, evaluating the results and documenting the deviations from expected results as defects.
- Using test administration or management and test monitoring tools as required.
- Automating tests (may be supported by a developer or a test automation expert).
- Where required, running the tests and measuring the performance of components and systems (if applicable).
- Reviewing tests developed by other testers.

If specialist testers are not available, then additional resources could be used at different test levels:

- For component and integration testing, any additional roles would typically be filled by someone from a development background.
- For system and user acceptance testing, any additional roles would typically be filled by someone from a business or user background.
- System operators (sometimes known as production support) would be responsible for operational acceptance testing.

As mentioned earlier, the thing to remember when looking at roles and tasks within a test project is that one person may have more than one role and carry out some or all of the tasks applicable to the role. This is different to having a 'job': a 'job' may contain many roles and tasks.

**CHECK OF UNDERSTANDING**

(1) What other names are given to the test leader role?

(2) Detail five possible tasks of a test leader.

(3) Detail five possible tasks of a tester.

(4) Describe the differences between a test leader role and a test leader task.

## TEST APPROACHES (TEST STRATEGIES)

A test approach is the implementation of a test strategy on a particular project. The test approach defines how testing will be implemented. A test approach can reflect testing for a whole organisation, a programme of work or an individual project. It can be:

- developed early in the life cycle, which is known as preventative – in this approach the test design process is initiated as early as possible in the life cycle to stop defects being built into the final solution;
- left until just before the start of test execution, which is known as reactive – this is where testing is the last development stage and is not started until after design and coding has been completed (sometimes it is identified as the waterfall approach, i.e. all development stages are sequential, the next not starting until the previous one has nearly finished).

A test approach includes all of the decisions made on how testing should be implemented, based upon the (test) project goals and objectives, as well as the risk assessment. It forms the starting point for test planning, selecting the test design techniques and test types to be employed. It should also define the software and test entry and exit criteria.

There are many approaches or strategies that can be employed. All will depend on the context within which the test team is working, and may consider risks, hazards and safety, available resources and skills, the technology, the nature of the system (e.g. custom built vs COTS), test objectives and regulations, and may include:

- Analytical approaches such as risk-based testing where testing is directed to areas of greatest risk (see later in this section for an overview of risk-based testing).
- Model-based approaches such as stochastic testing using statistical information about failure rates (such as reliability growth models) or usage (such as operational profiles).
- Methodical approaches, such as failure-based (including error guessing and fault attacks), checklist based and quality-characteristic based.
- Standard-compliant approaches, specified by industry-specific standards such as The Railway Signalling standards (which define the levels of testing required) or the MISRA (which defines how to design, build and test reliable software for the motor industry).
- Process-compliant approaches, which adhere to the processes developed for use with the various agile methodologies or traditional waterfall approaches.
- Dynamic and heuristic approaches, such as exploratory testing (see Chapter 4) where testing is more reactive to events than pre-planned, and where execution and evaluation are concurrent tasks.
- Consultative approaches, such as those where test coverage is driven primarily by the advice and guidance of technology and/or business domain experts outside or within the test team.
- Regression-averse approaches, such as those that include reuse of existing test material, extensive automation of functional regression tests, and standard test suites.

Different approaches may be combined if required. The decision as to how and why they will be combined will depend on the circumstances prevalent in a

project at the time. For example, an organisation may as a standard use an agile method, but in a particular situation the structure of the test effort could use a risk-based approach to ensure the testing is correctly focused.

Deciding on which approach to take may be dictated by standards, e.g. those used in the motor industry that are set by MISRA, or at the discretion of those developing the approach or strategy. Where discretion is allowed, the context or scenario needs to be taken into account. Therefore the following factors should be considered when defining the strategy or approach:

- Risk of failure of the project, hazards to the product and risks of product failure to humans, the environment and the company, e.g. the cost of failure would be too high (safety-critical environments).
- Skills and experience of the people in the proposed techniques, tools and methods. There is no point in implementing a sophisticated component-level, technique-driven approach or strategy when the only resources available are business users with no technical grounding.
- The objective of the testing endeavour and the mission of the testing team, e.g. if the objective is to find only the most serious defects.
- Regulatory aspects, such as external and internal regulations for the development process, e.g. The Railway Signalling standards that enforce a given level of test quality.
- The nature of the product and the business, e.g. a different approach is required for testing mobile phone coverage than for testing an online banking operation.

**CHECK OF UNDERSTANDING**

(1) Name and explain five approaches to the development of the test approach or test strategy.

(2) Name one of the standards referred to that dictate the test approach.

(3) Can discretion be used when defining a test approach and if so what can influence the decision as to which way to approach testing?

## TEST PLANNING AND ESTIMATION

### Test planning

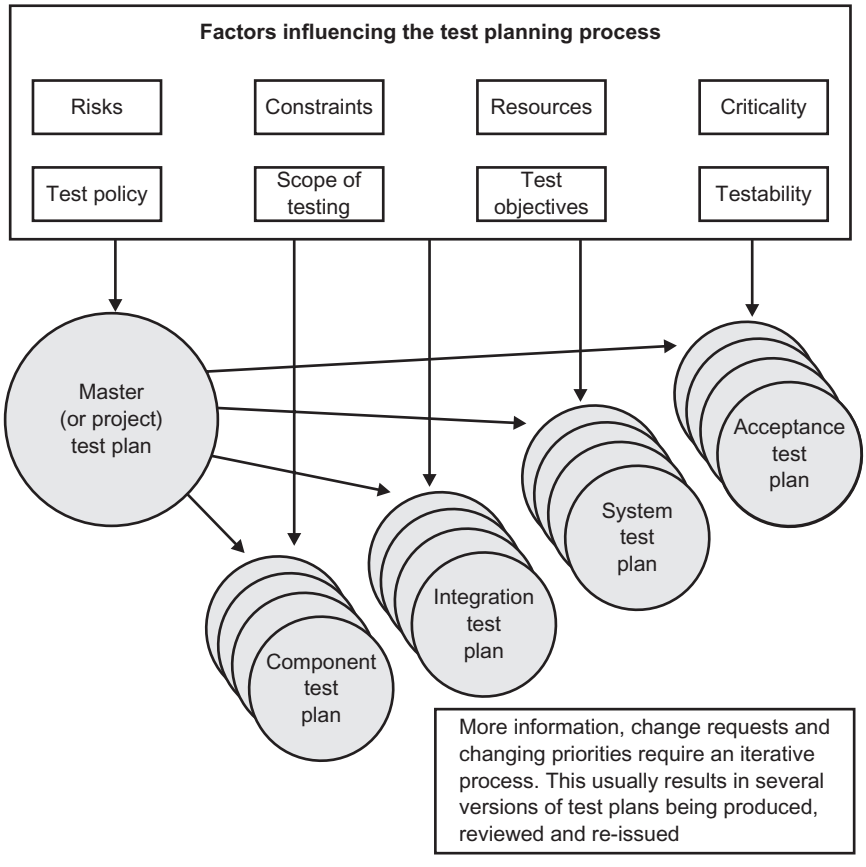
Test planning is the most important activity undertaken by a test leader in any test project. It ensures that there is initially a list of tasks and milestones in a baseline plan to track progress against, as well as defining the shape and size of the test effort. Test planning is used in development and implementation projects (sometimes called 'greenfield') as well as maintenance (change and fix) activities.

The main document produced in test planning is often called a master test plan or a project test plan. This document defines the high level of the test activities being planned. It is normally produced during the early phases of the project (e.g. initiation) and will provide sufficient information to enable a test project to be established (bearing in mind that at this point in a project little more than requirements may be available from which to plan).

The details of the test-level activities are documented within test-level plans, e.g. the system test plan. These documents will contain the detailed activities and estimates for the relevant test level.

Figure 5.2 shows where test-level test plans fit into the V-model. It shows how a test plan exists for each test level and that they will usually refer to the master test plan.

**Figure 5.2** Test plans in the V-model



The contents sections of a test plan for either the master test plan or test-level plans are normally identical or very similar. IEEE 829, the *Standard for Software Test Documentation*, contains details of what the content of the plans should be.

The IEEE 829 standard identifies that there should be a minimum of 16 sections present in a test plan, as in Table 5.2.

Test planning is a continual activity that spans the life of the test project; it takes place in all life-cycle stages. As risks and changes occur, the plan and planning should be amended to recognise these and reflect the current position. As the plans will have been baselined (locked down) after initial sign-off, these changes would normally be managed by the project change process. Baselining a document effectively secures it from further change unless authorised via a change control process.

**Table 5.2** Test plan sections

| Section no. | Heading | Details |
|---|---|---|
| 1 | Test plan identifier | A unique identifying reference such as<br>'Doc ref XYZ v2' |
| 2 | Introduction | A brief introduction to the document and the project for which it has been produced |
| 3 | Test items | A test item is a software item that is the object of testing<br>A software item is one or more items of source code, object code, job control code, or control data<br>This section should contain any documentation references, e.g. design documents |
| 4 | Features to be tested | A feature is a distinguishing characteristic of a software item (e.g. performance, portability, or functionality)<br>Identify all software features and combinations of features and the associated test design specification |
| 5 | Features not to be tested | Identify all software features and significant combinations and state the reasons for not including them |

*(Continued)*

**Table 5.2** *(Continued)*

| Section no. | Heading | Details |
|---|---|---|
| 6 | Approach | Details the overall approach to testing; this could include a detailed process definition, or could refer to other documentation where the detail is documented, i.e. a test strategy |
| 7 | Item pass/fail criteria | Used to determine whether a software item has passed or failed its test |
| 8 | Suspension and resumption requirements | Suspension requirements define criteria for stopping part or all of the testing activity Resumption requirements specify the requirements to resume testing |
| 9 | Test deliverables | The documents that testing will deliver, e.g. from IEEE 829 documents such as:<br>• test plans (for each test level)<br>• test specifications (design, case and procedure)<br>• test summary reports |
| 10 | Testing tasks | All tasks for planning and executing the testing, including the intertask dependencies |
| 11 | Environmental needs | Definition of all environmental requirements such as hardware, software, PCs, desks, stationery, etc. |
| 12 | Responsibilities | Identifies the roles and tasks to be used in the test project and who will own them |
| 13 | Staffing and training needs | Identifies any actual staffing requirements and any specific skills and training requirements, e.g. automation |
| 14 | Schedule | Document delivery dates and key milestones |
| 15 | Risks and contingencies | High-level project risks and assumptions and a contingency plan for each risk |
| 16 | Approvals | Identifies all approvers of the document, their titles and the date of signature |

A useful revision aid to help remember the 16 sections of the IEEE 829 test plan is the acronym 'SPACEDIRT', each letter mapping to one or several sections of the test plan:

**S** scope (including test items, features to be tested and features not to be tested)

**P** people (including responsibilities, staff and training and approvals)

**A** approach

**C** criteria (including item pass/fail criteria and suspension and resumption requirements)

**E** environment needs

**D** deliverables (test)

**I** identifier and introduction (test plan)

**R** risks and contingencies

**T** testing tasks and schedule

### Test-planning activities

During test planning various activities for an entire system or a part of a system have to be undertaken by those working on the plan. They include:

- Working with the project manager and subject matter experts' to determine the scope and the risks that need to be tested. As well identifying and agreeing the objectives of the testing, be they time, quality or cost focussed, or in fact maybe a mixture of all three. The objectives will enable the test project to know when it has finished.
- Putting together the overall approach of testing (sometimes called the test strategy), ensuring that the test levels and entry and exit criteria are defined.
- Liaising with the project manager and making sure that the testing activities have been included within the software life-cycle activities such as:
    - design – the development of the software design;
    - development – the building of the code;
    - implementation – the activities surrounding implementation into a live environment.
- Working with the project to decide what needs to be tested, what roles are involved and who will perform the test activities, planning when and how the test activities should be done, deciding how the test results will be evaluated, and defining when to stop testing (exit criteria).
- Building a plan that identifies when and who will undertake the test analysis and design activities. In addition to the analysis and design activities test planning should also document the schedule for test implementation, execution and evaluation.
- Finding and assigning resources for the different activities that have been defined.

- Deciding what the documentation for the test project will be, e.g. which plans, how the test cases will be documented, etc.
- Defining the management information, including the metrics required and putting in place the processes to monitor and control test preparation and execution, defect resolution and risk issues.
- Ensuring that the test documentation generates repeatable test assets, e.g. test cases.

### Entry criteria

Entry criteria are used to determine when a given test activity can start. This could include the beginning of a level of testing, when test design and/or when test execution is ready to start.

Examples of some typical entry criteria to test execution (for example) may include:

- Test environment available and ready for use (it functions).
- Test tools installed in the environment are ready for use.
- Testable code is available.
- All test data is available and correct.
- All test design activity has completed.

### Exit criteria

Exit criteria are used to determine when a given test activity has been completed or when it should stop. Exit criteria can be defined for all of the test activities, such as planning, specification and execution as a whole, or to a specific test level for test specification as well as execution.

Exit criteria should be included in the relevant test plans.

Some typical exit criteria might be:

- All tests planned have been run.
- A certain level of requirements coverage has been achieved.
- No high-priority or severe defects are left outstanding.
- All high-risk areas have been fully tested, with only minor residual risks left outstanding.
- Cost – when the budget has been spent.
- The schedule has been achieved, e.g. the release date has been reached and the product has to go live. This was the case with the millennium testing (it had to be completed before midnight on 31 December 1999), and is often the case with government legislation.

Exit criteria should have been agreed as early as possible in the life cycle; however, they can be and often are subject to controlled change as the detail of

the project becomes better understood and therefore the ability to meet the criteria is better understood by those responsible for delivery.

**CHECK OF UNDERSTANDING**

(1) What is the name of the international standard for test documentation?

(2) Identify the 16 sections of the test plan.

(3) What activities are contained within test planning?

(4) Detail four typical exit criteria.

### Test estimation

The syllabus details two test estimation approaches, metrics-based and expert based. The two approaches are quite different, the former being based upon data whilst the latter is a somewhat subjective approach.

### The metrics-based approach

This approach relies upon data collected from previous or similar projects. This kind of data might include:

- The number of test conditions.
- The number of test cases written.
- The number of test cases executed.
- The time taken to develop test cases.
- The time taken to run test cases.
- The number of defects found.
- The number of environment outages and how long on average each one lasted.

With this approach and data it is possible to estimate quite accurately what the cost and time required for a similar project would be.

It is important that the actual costs and time for testing are accurately recorded. These can then be used to revalidate and possibly update the metrics for use on the next similar project.

### The expert-based approach

This alternative approach to metrics is to use the experience of owners of the relevant tasks or experts to derive an estimate (this is also known as the Wide Band Delphi approach). In this context 'experts' could be:

- Business experts.
- Test process consultants.

- Developers.
- Technical architects.
- Analysts and designers.
- Anyone with knowledge of the application to be tested or the tasks involved in the process.

There are many ways that this approach could be used. Here are two examples:

- Distribute a requirement specification to the task owners and get them to estimate their task in isolation. Amalgamate the individual estimates when received; build in any required contingency, to arrive at the estimate.
- Distribute to known experts who develop their individual view of the overall estimate and then meet together to agree on and/or debate the estimate that will go forward.

Expert estimating can use either of the above approaches individually or mixing and matching them as required.

Many things affect the level of effort required to fulfil the test requirements of a project. These can be split into three main categories, as shown below.

- Product characteristics:
  - size of the test basis;
  - complexity of the final product;
  - the amount of non-functional requirements;
  - the security requirements (perhaps meeting BS 7799, the security standard);
  - how much documentation is required (e.g. some legislation-driven changes demand a certain level of documentation that may be more than an organisation would normally produce);
  - the availability and quality of the test basis (e.g. requirements and specifications).
- Development process characteristics:
  - timescales;
  - amount of budget available;
  - skills of those involved in the testing and development activity (the lower the skill level in development, the more defects could be introduced, and the lower the skill level in testing, the more detailed the test documentation needs to be);

  - which tools are being used across the life cycle (i.e. the amount of automated testing will affect the effort required).

- Expected outcome of testing such as:
  - the amount of errors;
  - test cases to be written.

Taking all of this into account, once the estimate is developed and agreed the test leader can set about identifying the required resources and building the detailed plan.

**CHECK OF UNDERSTANDING**

(1) Compare and contrast the two approaches to developing estimates.

(2) Provide three examples of what a metrics approach to estimates would use as a base.

(3) Name three areas that affect the level of effort to complete the test activity.

## TEST PROGRESS MONITORING AND CONTROL

### Test progress monitoring

Having developed the test plan, the activities and timescales determined within it need to be constantly reviewed against what is actually happening. This is test progress monitoring. The purpose of test progress monitoring is to provide feedback and visibility of the progress of test activities.

The data required to monitor progress can be collected manually, e.g. counting test cases developed at the end of each day, or, with the advent of sophisticated test management tools, it also possible to collect the data as an automatic output from a tool either already formatted into a report, or as a data file that can be manipulated to present a picture of progress.

The progress data is also used to measure exit criteria such as test coverage, e.g. 50 per cent requirements coverage achieved.

Common test metrics include:

- Percentage of work done in test case preparation (or percentage of planned test cases prepared).
- Percentage of work done in test environment preparation.
- Test case execution (e.g. number of test cases run/not run, and test cases passed/failed).

- Defect information (e.g. defect density, defects found and fixed, failure rate and retest results).
- Test coverage of requirements, risks or code.
- Subjective confidence of testers in the product.
- Dates of test milestones.
- Testing costs, including the cost compared with the benefit of finding the next defect or to run the next test.

Ultimately test metrics are used to track progress towards the completion of testing, which is determined by the exit criteria. So test metrics should relate directly to the exit criteria.

There is a trend towards 'dashboards', which reflect all of the relevant metrics on a single screen or page, ensuring maximum impact. For a dashboard, and generally when delivering metrics, it is best to use a relatively small but impact-worthy subset of the various metric options available. This is because the readers do not want to wade through lots of data for the key item of information they are after, which invariably is 'Are we on target to complete on time?'

These metrics are often displayed in graphical form, examples of which are shown in Figure 5.3. This reflects progress on the running of test cases and reports on defects found. There is also a box at the top left for some written commentary on progress to be documented (this could simply be the issues and/or successes of the previous reporting period).

The graph in Figure 5.4 is the one shown at the bottom left of the dashboard in Figure 5.3. It reports the number of incidents raised, and also shows the planned and actual numbers of incidents.

**Figure 5.3** iTesting Executive Dashboard

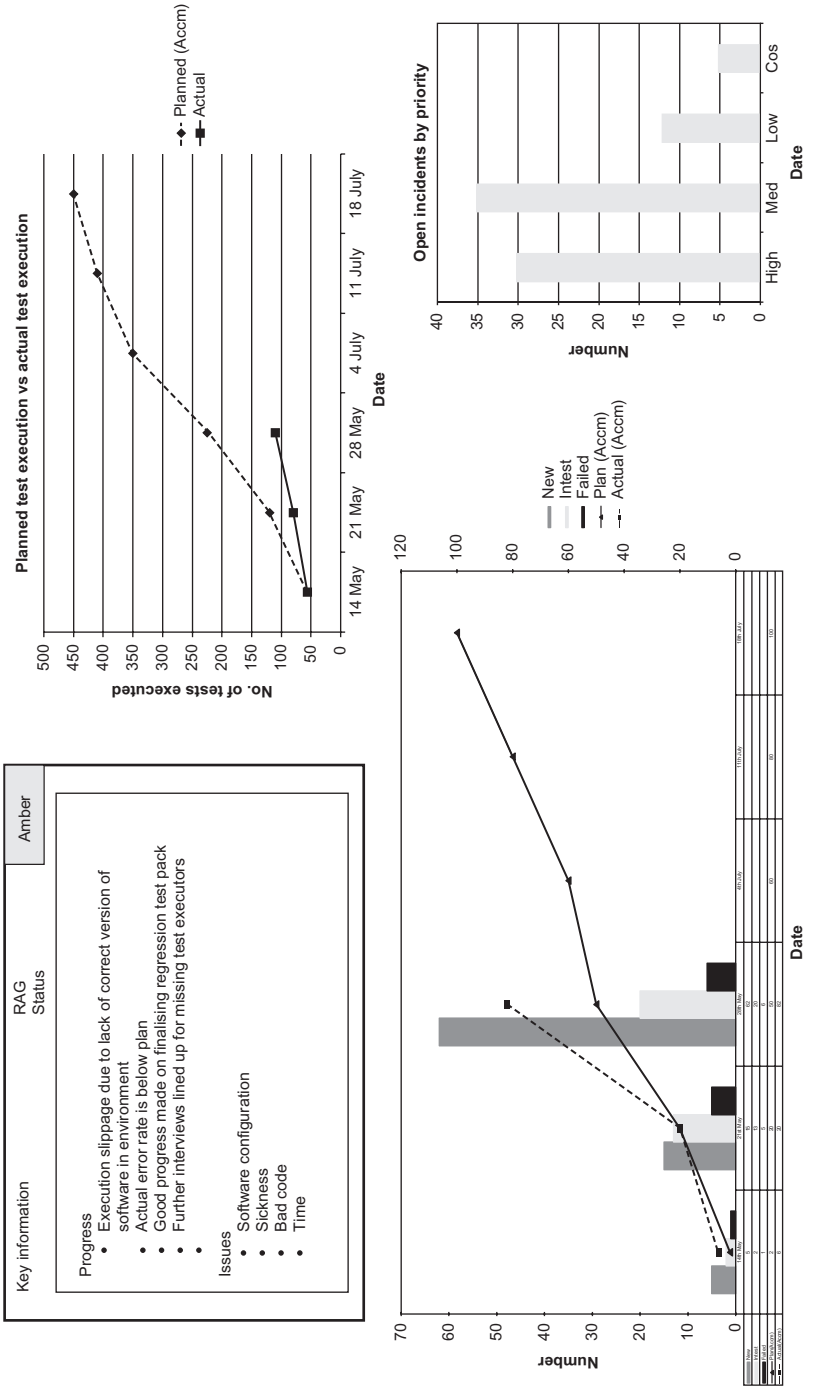
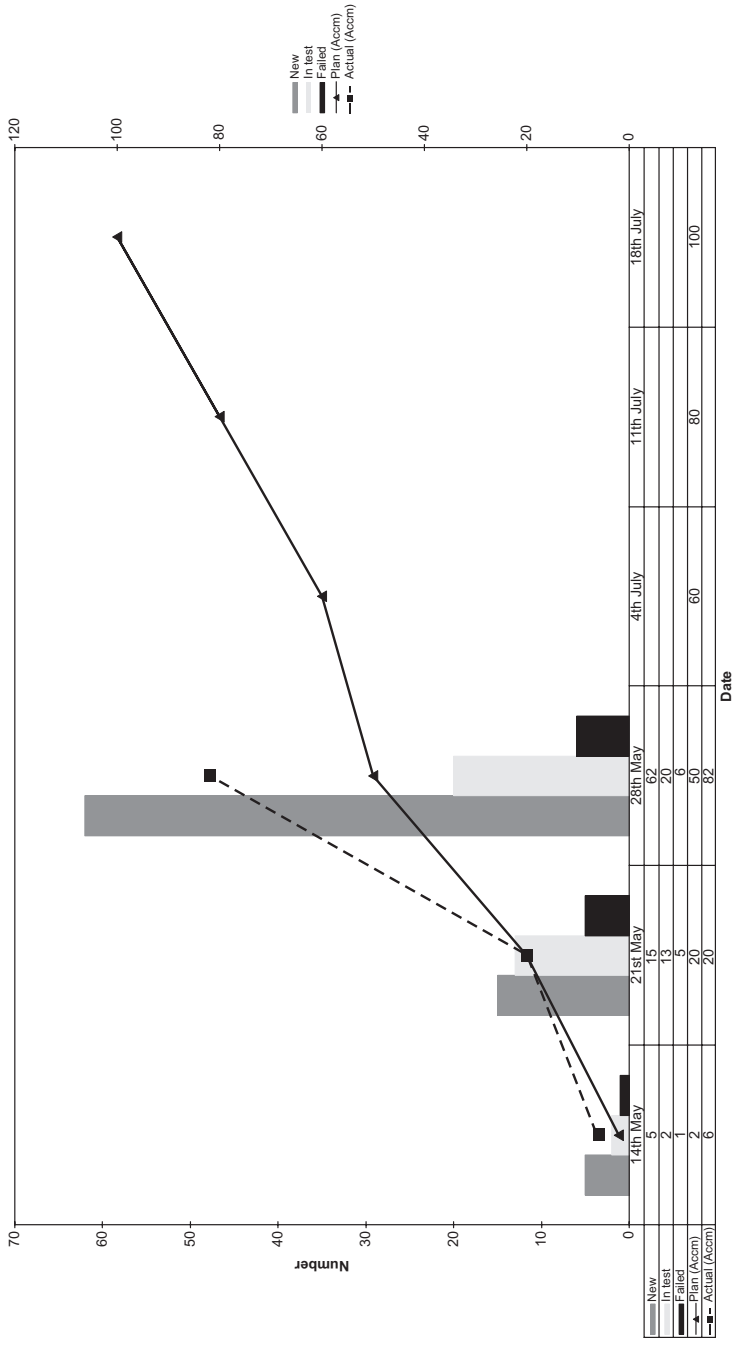


**Figure 5.4** Incidents planned/raised



**Test reporting**
Test reporting is the process where by test metrics are reported in summarised format to update the reader regarding the testing tasks undertaken. The information reported can include:

- What has happened during a given period of time, e.g. a week, a test level or the whole test endeavour, or when exit criteria have been met.
- Analysed information and metrics required to support recommendations and decisions about future actions, such as:
  - an assessment of defects remaining;
  - the economic benefit of continued testing, e.g. additional tests are exponentially more expensive than the benefit of running;
  - outstanding risks;
  - the level of confidence in tested software, e.g. defects planned vs actual defects found.

The IEEE 829 standard includes an outline of a test summary report that could be used for test reporting. The structure defined in the outline is shown in Table 5.3.

**Table 5.3** Test summary report outline

| Section no. | Heading | Details |
|---|---|---|
| 1 | Test summary report identifier | The specific identifier allocated to this document, e.g. TSR XYX v1 |
| 2 | Summary | Identifies the items tested (including any version numbers)<br>Documents the environments in which the test activity being reported on took place<br>References the testing documentation for each test item, e.g. test plan, test cases, test defect reports |
| 3 | Variances | Reports deviations from the test approach or strategy, test plan, test specification or test procedures |

*(Continued)*

**Table 5.3** *(Continued)*

| Section no. | Heading | Details |
|---|---|---|
| 4 | Comprehensive assessment | Measures the actual progress made against the exit criteria and explains why any differences have arisen |
| 5 | Summary results | Provides an overview of the results from the test activities; it should include details of defects raised and fixed, as well as those that remain unresolved |
| 6 | Evaluation | Provides an evaluation of the quality of each test item, including a view of the risks of failure in production of these test items. Based upon the test result metrics and test item pass/fail criteria |
| 7 | Summary of activities | A summary of the major test activities and events such as test environment unavailability, success or weaknesses of the test specification process, etc.<br>Should also include resource usage data, e.g. planned spend against actual spend |
| 8 | Approvals | Identifies all approvers of the document |

The information gathered can also be used to help with any process improvement opportunities. This information could be used to assess whether:

- the goals for testing were correctly set (were they achievable; if not why not?);
- the test approach or strategy was adequate (e.g. did it ensure there was enough coverage?);
- the testing was effective in ensuring that the objectives of testing were met.

**Test control**

We have referred above to the collection and reporting of progress data. Test control uses this information to decide on a course of action to ensure control of the test activities is maintained and exit criteria are met. This is particularly required when the planned test activities are behind schedule. The actions taken could impact any of the test activities and may also affect other software life-cycle activities.

Examples of test-control activities are as follows:

- Making decisions based on information from test monitoring.
- Reprioritise tests when an identified project risk occurs (e.g. software delivered late).
- Change the test schedule due to availability of a test environment.
- Set an entry criterion requiring fixes to be retested (confirmation tested) by a developer before accepting them into a build (this is particularly useful when defect fixes continually fail again when retested).
- Review of product risks and perhaps changing the risk ratings to meet the target.
- Adjusting the scope of the testing (perhaps the amount of tests to be run) to manage the testing of late change requests.

The following test-control activities are likely to be outside the test leader's responsibility. However, this should not stop the test leader making a recommendation to the project manager.

- Descoping of functionality, i.e. removing some less important planned deliverables from the initial delivered solution to reduce the time and effort required to achieve that solution.
- Delaying release into the production environment until exit criteria have been met.
- Continuing testing after delivery into the production environment so that defects are found before they occur in production.

**CHECK OF UNDERSTANDING**

(1) Name four common test metrics.

(2) Name the eight headings in the IEEE 829 summary report.

(3) Identify three ways a test leader can control testing if there are more tests than there is time to complete.

## INCIDENT MANAGEMENT

An incident is any unplanned event occurring that requires further investigation. In testing this translates into anything where the actual result is different to the expected result. An incident when investigated may be a defect, however, it may also be a change to a specification or an issue with the test being run. It is important that a process exists to track all incidents through to closure.

Incidents can be raised at any time throughout the software development life cycle, from reviews of the test basis (requirements, specifications, etc.) to test specification and test execution.

Incident management, according to IEEE 1044 (*Standard Classification for Software Anomalies*), is 'The process of recognising, investigating, taking action and disposing of incidents.' It involves recording incidents, classifying them and identifying the impact. The process of incident management ensures that incidents are tracked from recognition to correction, and finally through retest and closure. It is important that organisations document their incident management process and ensure they have appointed someone (often called an incident manager/coordinator) to manage/police the process.

Incidents are raised on incident reports, either electronically via an incident management system (from Microsoft Excel to sophisticated incident management tools) or on paper. Incident reports have the following objectives:

- To provide developers and other parties with feedback on the problem to enable identification, isolation and correction as necessary. It must be remembered that most developers and other parties who will correct the defect or clear up any confusion will not be present at the point of identification, so without full and concise information they will be unable to understand the problem, and possibly therefore unable to understand how to go about fixing it. The more information provided, the better.
- To provide test leaders with a means of tracking the quality of the system under test and the progress of the testing. One of the key metrics used to measure progress is a view of how many incidents are raised, their priority and finally that they have been corrected and signed off.
- To provide ideas for test process improvement. For each incident the point of injection should be documented, e.g. a defect in requirements or code, and subsequent process improvement can focus on that particular area to stop the same defect occurring again.

The details that are normally included on an incident report are:

- Date of issue, issuing organisation, author, approvals and status.
- Scope, severity and priority of the incident.
- References, including the identity of the test case specification that revealed the problem.
- Expected and actual results.
- Date the incident was discovered.
- Identification of the test item (configuration item) and environment.
- Software or system life-cycle process in which the incident was observed.
- Description of the incident to enable reproduction and resolution, including logs, database dumps or screenshots.

- Degree of impact on stakeholder(s) interests.
- Severity of the impact on the system.
- Urgency/priority to fix.
- Status of the incident (e.g. open, deferred, duplicate, waiting to be fixed, fixed awaiting confirmation test or closed).
- Conclusions, recommendations and approvals.
- Global issues, such as other areas that may be affected by a change resulting from the incident.
- Change history, such as the sequence of actions taken by project team members with respect to the incident to isolate, repair and confirm it as fixed.

The syllabus also recognises that IEEE 829 contains the outline of a test incident report. The outline suggests that the report should contain the sections shown in Table 5.4.

**Table 5.4** Test incident report outline

| Section no. | Heading | Details |
|---|---|---|
| 1 | Test incident report identifier | The unique identifier assigned to this test incident report |
| 2 | Summary | A summary of the incident, detailing where expected and actual results differ, identifying at a high level the items that are affected, and the steps leading up to the recognition of the incident, e.g. if in test execution, which test was executed and the actual keystrokes and data loaded |
| 3 | Incident description | A detailed description of the incident, which should include:<br>• Inputs<br>• Expected results<br>• Actual results<br>• Anomalies<br>• Date and time<br>• Procedure step |

*(Continued)*

**Table 5.4** *(Continued)*

| Section no. | Heading | Details |
|---|---|---|
| | | • Environment<br>• Attempts to repeat<br>• Testers' comments<br>• Observers' comments<br>Should also include any information regarding possible causes and solutions |
| 4 | Impact | If known, document what impact the incident has on progress |

**CHECK OF UNDERSTANDING**

(1) Identify three details that are usually included in an incident report.

(2) What is the name of the standard that includes an outline of a test incident report?

(3) What is a test incident?

## CONFIGURATION MANAGEMENT

Configuration management is the process of managing products, facilities and processes by managing the information about them, including changes, and ensuring they are what they are supposed to be in every case.

For testing, configuration management will involve controlling both the versions of code to be tested and the documents used during the development process, e.g. requirements, design and plans.

In both instances configuration management should ensure traceability throughout the test process, e.g. a requirement should be traceable through to the test cases that are run to test its levels of quality, and vice versa.

Effective configuration management is important for the test process as the contents of each release of software into a test environment must be understood and at the correct version, otherwise testers could end up wasting time because

either they are testing an invalid release of the software or the release does not integrate successfully, leading to the failure of many tests.

In most instances the project will have already established its configuration management processes that will define the documents and code to be held under configuration management. If this is not the case then during test planning the process and tools required to establish the right configuration management processes will need to be selected/implemented by the test leader.

The same principle applies to testware. Each item of testware (such as a test procedure) should have its own version number and be linked to the version of the software it was used to test. For example, test procedure TP123a might be used for Release A and TP123b might be used for Release B – even though both have the same purpose and even expected results. However, another test procedure, TP201, may be applicable to all releases.

A good configuration management system will ensure that the testers can identify exactly what code they are testing, as well as have control over the test documentation such as test plans, test specification, defect logs, etc.

### CHECK OF UNDERSTANDING

(1) Define configuration management.

(2) What can be stored under configuration management?

(3) Why is it important to have effective configuration management?

## SUMMARY

In this chapter we have looked at the component parts of test management. We initially explored risk and testing. When developing the test plan, the test leader and tester will look at the product risks (risks that relate directly to the failure of the product in the live environment) to decide what is important to test, as well as ensuring that any project risks (risks relating to the delivery of the project) are mitigated.

The importance of independence in the test organisation and how independence helps to ensure that the right focus is given to the test activity was reviewed. Independence is gained by separating the creative development activity from the test activity and we looked at the different levels of independence that are achievable:

- The developers – low independence.
- Independent testers ceded to the development team.

- Independent permanent test team, centre of excellence with the organisation.
- Independent testers or test team provided by the operational business unit.
- Specialist testers such as security testers or performance testers.
- Outsourced test team or the use of independent contractors – high independence.

We have looked at the test approach and how it shapes the test activity based on many influences, including risks and the objectives of the testing.

We have reviewed two roles that exist within a test project, test leader (also known as a test manager or test coordinator) and tester. Both roles are important to the delivery of testing, but could be vested in one or many people, e.g. one person could have the role of test manager and tester. A test leader has responsibility for all of the planning activity, whilst the tester has responsibility for activities that surround the preparation of test cases.

IEEE 829, the test documentation standard, provides outlines of three test planning documents:

- The test plan
- The test summary report
- The test incident report

Test management depends not only on the preparation of the required documents but also on the development of the right entry and exit criteria and estimates, the monitoring of progress through the plan and the control activities implemented to ensure the plan is achieved.

Test estimating can be achieved in one of two ways: using historical metrics or the expert-based approach (involving experts in the subject who agree the estimate between them).

After a plan of activity has been developed and time begins to pass the test leader needs to monitor the progress of the activities. If any activity is delayed or there has been a change of any kind in the project itself, the test leader may need to revise the plan or take other actions to ensure the project is delivered on time.

We explored how the incidents found during testing are recorded, and we reviewed the level of detail that needs to be recorded to ensure that any defect is fully understood and that any fix then made is the right one.

Finally we looked at configuration management. When running test cases against the code it is important that the tester is aware of the version of code being tested

and the version of the test being run. Controlling the versioning of the software and test assets is called configuration management. Lack of configuration management may lead to issues like loss of already-delivered functionality, reappearance of previously corrected errors and no understanding of which version of test was run against which version of code.

## Example examination questions with answers

### E1. K1 question

When assembling a test team to work on an enhancement to an existing system, which of the following has the highest level of test independence?

a. A business analyst who wrote the original requirements for the system.
b. A permanent programmer who reviewed some of the new code, but has not written any of it.
c. A permanent tester who found most defects in the original system.
d. A contract tester who has never worked for the organisation before.

### E2. K2 question

What test roles (or parts in the testing process) is a developer **most likely** to perform?

(i) Executing component integration tests.
(ii) Static analysis.
(iii) Setting up the test environment.
(iv) Deciding how much testing should be automated.

a. (i) and (ii)
b. (i) and (iv)
c. (ii) and (iii)
d. (iii) and (iv)

### E3. K2 question

Which of the following are valid justifications for developers testing their own code during unit testing?

(i) Their lack of independence is mitigated by independent testing during system and acceptance testing.
(ii) A person with a good understanding of the code can find more defects more quickly using white-box techniques.
(iii) Developers have a better understanding of the requirements than testers.
(iv) Testers write unnecessary incident reports because they find minor differences between the way in which the system behaves and the way in which it is specified to work.

a. (i) and (ii)
b. (i) and (iv)
c. (ii) and (iii)
d. (iii) and (iv)

### E4. K1 question

Which of the following terms is used to describe the management of software components comprising an integrated system?

a. Configuration management
b. Incident management
c. Test monitoring
d. Risk management

### E5.

A new system is about to be developed. Which of the following functions has the **highest** level of risk?

a. likelihood of failure = 20%; impact value = £100,000
b. likelihood of failure = 10%; impact value = £150,000
c. likelihood of failure = 1%; impact value = £500,000
d. likelihood of failure = 2%; impact value = £200,000

### E6.

Which of the following statements about risks is most accurate?

a. Project risks rarely affect product risk.
b. Product risks rarely affect project risk.
c. A risk-based approach is more likely to be used to mitigate product rather than project risks.
d. A risk-based approach is more likely to be used to mitigate project rather than product risks.

### Answers to questions in the chapter

**SA1.** The correct answer is c.
**SA2.** The correct answer is d.
**SA3.** The correct answer is a.

### Answers to example questions

**E1.** The correct answer is d.
In this scenario, the contract tester who has never worked for the organisation before has the highest level of test independence. The three others are less independent as they are likely to make assumptions based on their previous knowledge of the requirements, code and general functionality of the original system.

Note that independence does not necessarily equate to most useful. In practice most test or project managers would recruit a permanent tester who has worked on the original system in preference to a contract tester with no knowledge of the system. However, when assembling a team it would be useful to have staff with varying levels of test independence and system knowledge.

**E2.** The correct answer is a.

(i) Executing component integration tests is usually done by developers. Developers are usually responsible for unit and component integration testing. Independent testing usually follows at system and acceptance test levels.

(ii) Static analysis is usually done by developers because: it requires an understanding of the code and therefore the person doing this needs skills in the programming language; and it can be done as soon as the code is written. Therefore it is quick and effective for the developer to do it. The risk of a lack of test independence can be mitigated by performing independent system and acceptance testing.

(iii) Setting up the test environment is an activity typically performed by a tester. It may require support from developers and staff from other departments and on some occasions environments could be set up by developers. However, it is a task that could be done by a tester rather than a developer.

(iv) Deciding how much testing should be automated is typically a decision made by the test leader, who will consult other staff in the decision-making process. Developers may be involved and their skills may be required to automate some tests. However, the decision on how much to automate should not be made by developers.

**E3.** The correct answer is a.
It is unlikely that developers will have a better understanding of the requirements than testers, partly because testers work closely with the user community (and may be drawn from it) and partly because developers seldom work with the complete set of requirements in a medium to large development.

Testers may raise incidents related to the difference between user expectations and the specification, but these are not unnecessary. Such issues are more likely to arise at the later stages of testing.

Early testing (unit testing) is usually done most effectively by developers with a good understanding of the code and the development environment; they can be more efficient and more effective at this level. Later independent stages of testing offset any disadvantage from the lack of independence at unit testing level.

**E4.** The correct answer is a.
Incident management is the collection and processing of incidents raised when errors and defects are discovered. Test monitoring identifies the status of the testing activity on a continuous basis. Risk management identifies, analyses and mitigates risks to the project and the product. Configuration management is concerned with the management of changes to software components and their associated documentation and testware.

**E5.** The correct answer is a.
In (b) the product of probability × impact has the value £15,000; in (c) the value is £5,000 and in (d) it is £4,000. The value of £20,000 in (a) is therefore the highest.

**E6.** The correct answer is c.
In general, project risk and product risk can be hard to differentiate. Anything that impacts on the quality of the delivered system is likely to lead to delays or increased costs as the problem is tackled. Anything causing delays to the project is likely to threaten the delivered system's quality. The risk-based approach is an approach to managing product risk through testing, so it impacts most directly on product risk.